

**King Fahd University of Petroleum and Minerals
College of Computer Sciences and Engineering
Systems Engineering Department**

**CISE 422: Intelligent Controllers
(Term 191)**

Fuzzy Based Obstacle Avoidance Robot Using Raspberry Pi Microcontroller

Advisor: Dr. Mohammed Mysorewala

Bin Mohaya, Turki	201569090
Ali Goradi	201440400
Ibrahim Ali	201472500

December 11, 2019

INTRODUCTION

Nowadays, the world is moving toward the empowerment of robotics in different functions and areas to replace labor work jobs or solve complex issues (e.g. surgical robotics). In this project, we present an attempt to model an obstacle avoidance robot that can detect its environment and accordingly take action to move precisely. This attempt is done through the application of Artificial Intelligence (mainly fuzzy logic control). In this report, we cover the hardware part first, and then we discuss the implemented algorithm on the microcontroller. Afterthought, we overgo an analysis section to understand more the nature of such robotics.

HARDWARE COMPONENTS

To accomplish the goal, the following list of components were used:

- Ultrasonic Distance Sensor (HC-SR04)
- Light 5V DC Motors
- Raspberry Pi 4 Microcontroller
- H-bridge L298 (Amplifier)
- PQ1 Power Bank with 10,000 mAh

The overall assembly of the robot is shown in the next two figures:

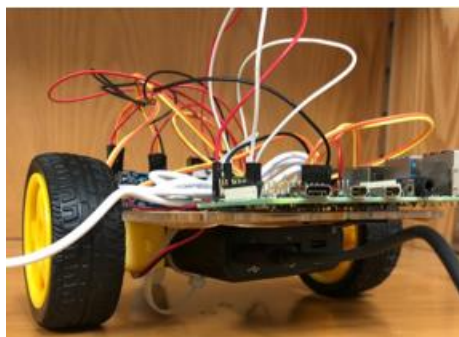


Figure 1: Backward Picture of The Robot

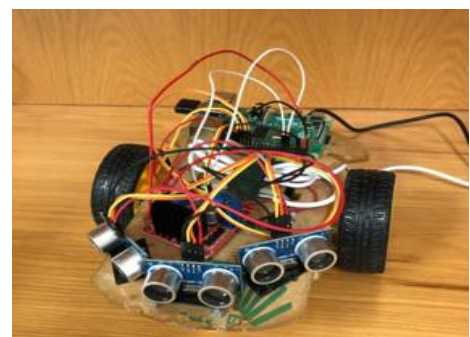


Figure 2: Forward Picture of The Robot

INPUTS

There are three inputs to the system where each sensor provides a single input signal. These signals are, then, fed into a signal conditioning algorithm to calculate the estimated distance between the robot and the next obstacle. The concept behind it is smooth and can be easily understood. To trigger the ultrasonic sensor a pulse signal of width 10 micro-second is generated from the microcontroller and a timer starts counting. Once this signal is received, the timer stops and multiplies half of the time calculated by the ultrasonic wave speed in air **343 m/s** as the following formula illustrates (half is multiplied only because the signal reflects and gets back).

$$d = \frac{t}{2} v = \frac{343}{2} t = 171.5t \text{ (m)}$$

This is done, for all the three sensors, simultaneously, then the three results are sent to the controller block to calculate the appropriate fuzzy output which we cover in the next part.

OUTPUTS

Only two outputs are generated from this process; the speed of each motor independently. To control direction and stability, different speeds are produced to each motor as the fuzzy controller calculates.

Through the following block diagram, we can follow the steps of the control loop and the sequence in which processes take place.

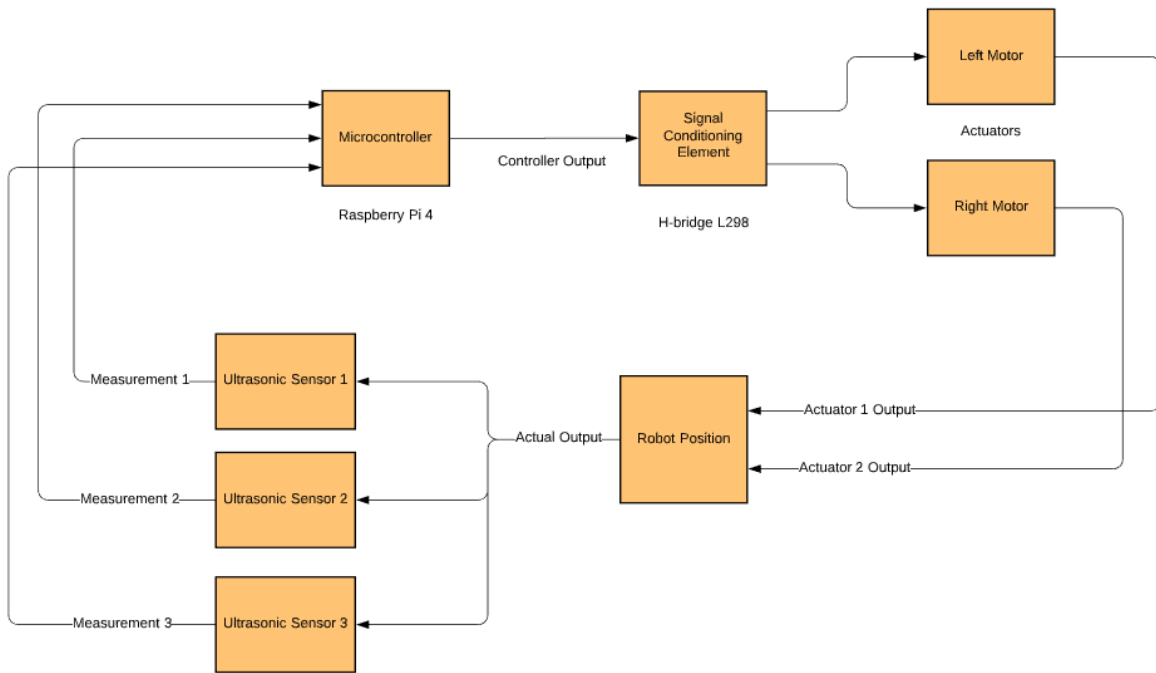


Figure 3: Control Block Diagram

As described above, the process overgoes three main stages; monitoring, comparing and taking action. The monitoring stage is handled by the Ultrasonic sensors and the signal conditioning algorithm implemented in the microcontroller. The comparing part is processed through the Raspberry Pi (Although there is no actual set point to compare with, there exists an implicit set point in which all obstacles should be far). Taking an action step is next occurring when the controller output is amplified through the H-bridge to the two DC motors and so on.

CIRCUIT DIAGRAM

To visualize how all the components are assembled, we present the following circuit diagram of the system drawn using the Fritzing software.

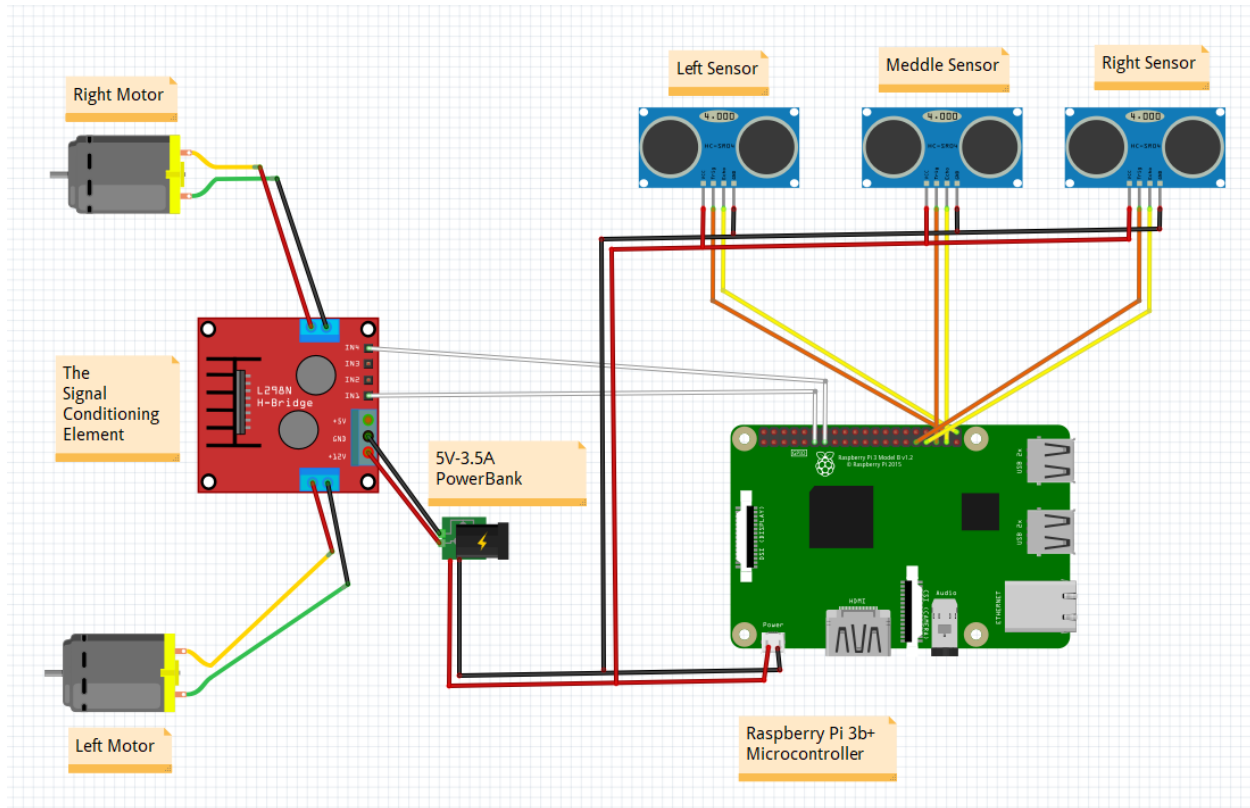


Figure 4: The Complete Circuit Diagram

SOFTWARE ALGORITHMS

We can divide the software part of the project into three stages. The first stage is related to the measurement signal conditioning in which the microcontroller takes several measurements and shoot for the best estimator before sending it to the fuzzy block. The second stage is related to the complete fuzzy control process. Here, the microcontroller starts calculating Degrees of Fulfillment, Implication, Inference, and finally Defuzzification. The last stage considers recording input and output data using Pandas data frame for further analysis and study. We consider each stage separately in the following discussion.

INPUT SIGNAL CONDITIONING

Since the repeatability of the Ultrasonic sensors is quite low, we developed an algorithm to tackle this issue. The objective of this algorithm is to provide the nearest possible estimator of the actual measurement by repeating it several times. The following figure visualizes this concept.

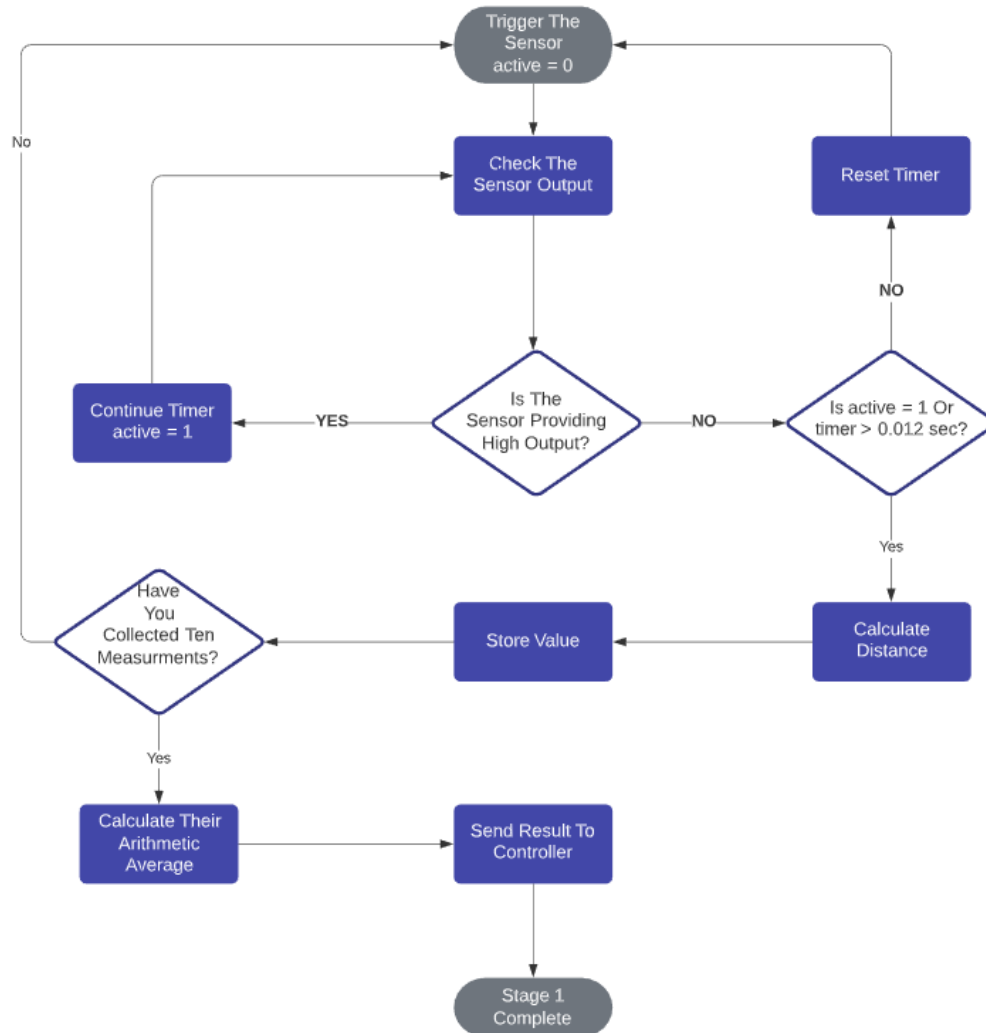


Figure 5: Measurement Signal Conditioning Algorithm

THE FUZZY CONTROLLER

In this system, we implemented a fuzzy logic controller with Mamdani model using MATLAB Fuzzy Toolbox. In the general structure of the controller, we defined three fuzzy inputs and two fuzzy outputs. The following figure illustrates the general controller structure.

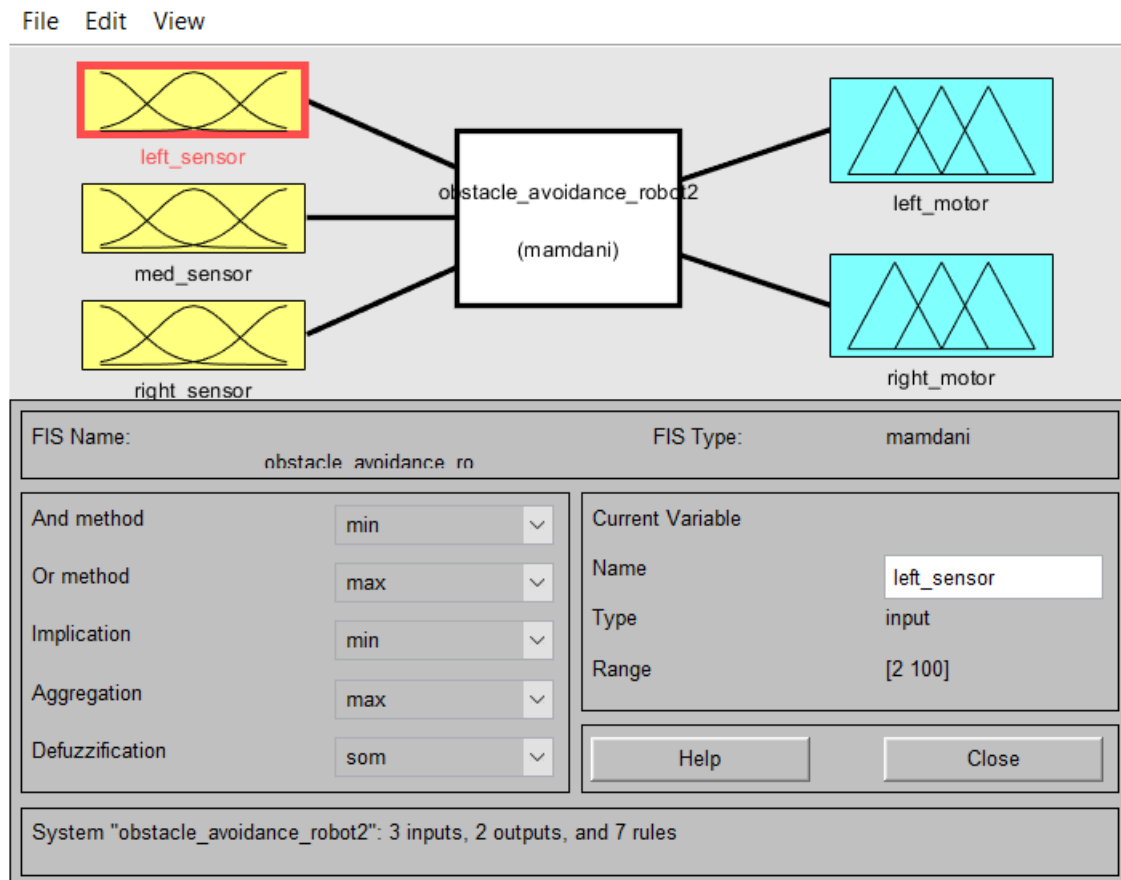


Figure 6: General Controller Structure

For each input, two membership functions were defined; CLOSE and FAR. Since the sensor range is between 2 - 400 cm, we defined the ranges as the following figure shows:

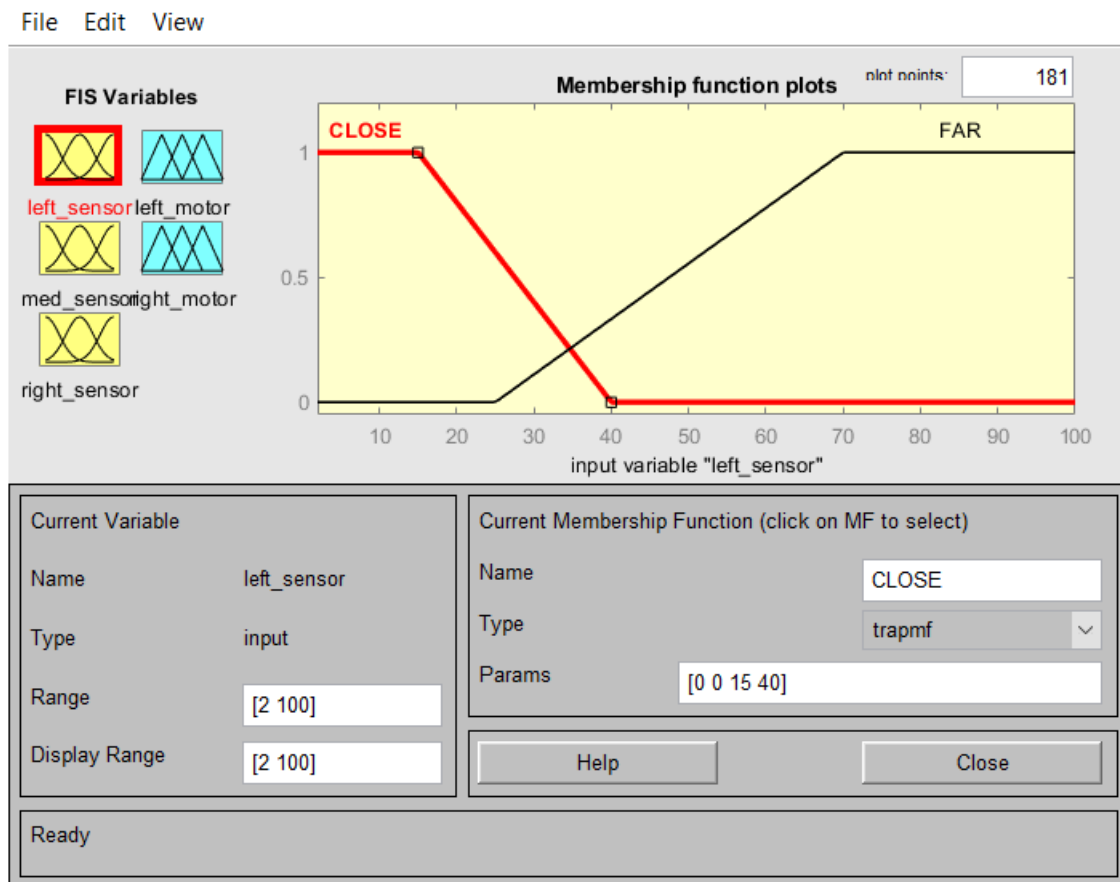


Figure 7: Input Membership Functions and Their Ranges

For the two outputs, which represent the motors' speeds in terms of voltage, we defined three membership functions as SLOW, MED, FAST. The following figure explains the output membership functions.

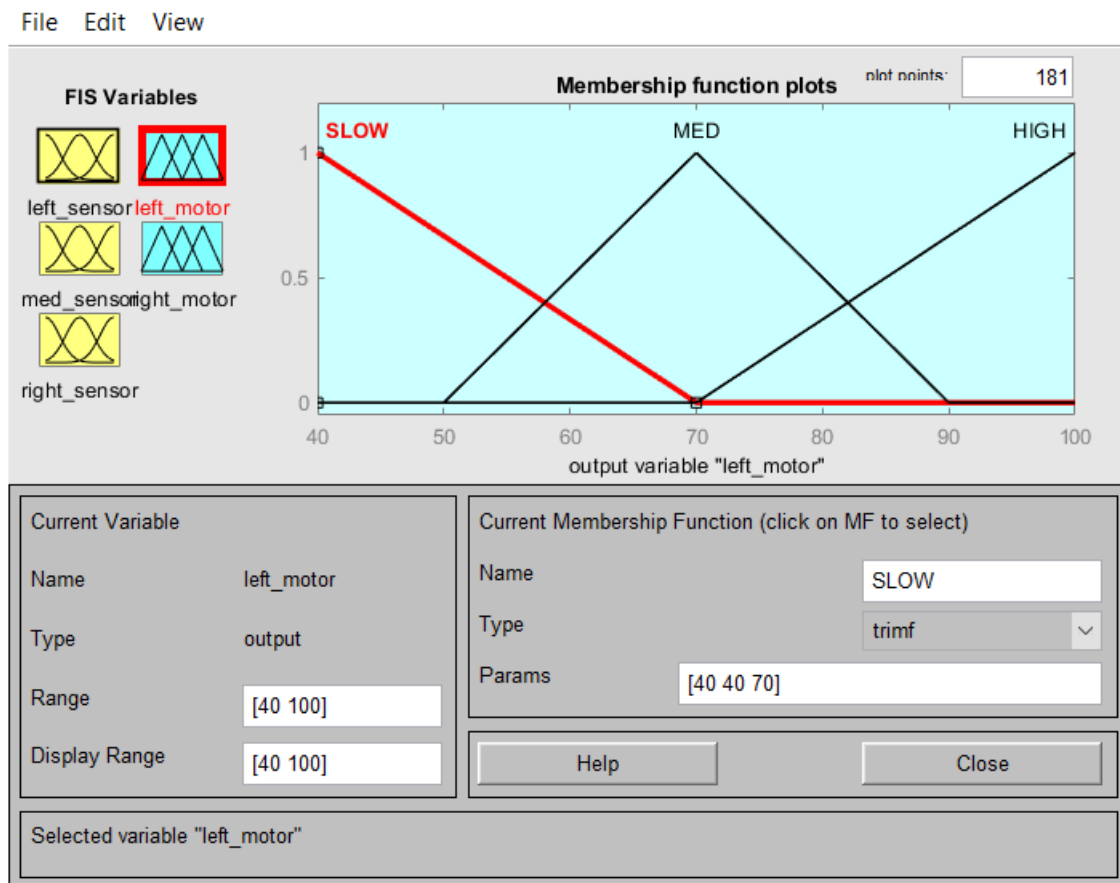


Figure 8: Output Membership Functions and Their Ranges

The x-axis of the output represents the duty cycle of the Pulse Width Modulation Signal provided directly by the microcontroller. The reason we started from a 40% duty cycle rather than 0% is that the motor cannot overcome static friction force unless provided with a minimum 40% duty cycle (Calculated by trial and error).

Now that we have present the general structure as well as the input and output membership functions, we designed seven rules to accomplish the objective. the following table illustrates the designed rules:

Left Sensor	Meddle Sensor	Right Sensor	Left Motor	Right Motor
CLOSE	CLOSE	FAR	FAST	SLOW
CLOSE	FAR	CLOSE	MED	MED
FAR	CLOSE	CLOSE	SLOW	FAST
CLOSE	FAR	FAR	MED	SLOW
FAR	CLOSE	FAR	SLOW	FAST
FAR	FAR	CLOSE	SLOW	MED
FAR	FAR	FAR	FAST	FAST

Table 1: Controller Fuzzy Rules

By connecting all these stages, we were able to accomplish our objective using the soft computing principles without precise mathematical modeling of mechanical systems. By doing that we generated 1600 real samples using Pandas data frame module in Python. The following section of the report represent a further analysis of the data.

STATISTICAL ANALYSIS

We present in this section, a statistical analysis using Anaconda Jupyter Notebook Tool that runs on Python. The following figure shows a sample from the data gathered.

	left_sensor	med_sensor	right_sensor	sampling_time	left_motor	right_motor
720	23.02	23.04	23.66	0.048108	1.966973	2.184353
148	23.93	22.49	14.57	0.044242	1.312214	2.678941
1006	48.21	40.10	33.89	0.078700	3.036000	3.036000
1258	20.63	23.06	360.33	0.242861	3.018386	1.003873
1415	25.97	30.70	26.04	0.055759	2.164822	2.160168

Table 2: Sample Data

As shown above, the data contain five variables, left_sensor, med_sensor, and right_sensor are the three measurement variables. The sampling_time column represents the amount of time required to complete one full loop in the control process (monitoring, comparing, and taking action). The last two columns

left_motor and right_motor resembles the two microcontroller output voltages fed to the H-bridge.

To understand the data more, we provide the following general statistical measures:

	left_sensor	med_sensor	right_sensor	sampling_time	left_motor	right_motor
count	1647.000000	1647.000000	1647.000000	1647.000000	1647.000000	1647.000000
mean	34.290407	40.392860	60.947286	0.087118	2.270863	1.881868
std	69.002414	81.121437	130.184738	0.111253	0.790695	0.850627
min	0.220000	0.130000	1.280000	0.016199	0.979000	0.979000
25%	13.595000	13.590000	19.725000	0.037895	1.541750	0.979000
50%	20.660000	20.260000	25.290000	0.048554	2.470326	1.810925
75%	27.625000	26.900000	32.060000	0.065490	3.036000	3.036000
max	470.160000	457.420000	676.890000	0.941171	3.036000	3.036000

Table 3: General Statistical Estimators

We can see that the dataset contains exactly 1647 samples of 5 variables. The left motor mean is higher than the right motor which contrast the rules (Three rules activate right motor as fast). This may be the case due to the shape of the test path. The sampling time is very high; 87 ms. We believe it is such due to the input signal conditioning algorithm described in figure 4. Overall, the general structure of our dataset is clean and realistic.

Let us examine the variables distributions in the next figure.

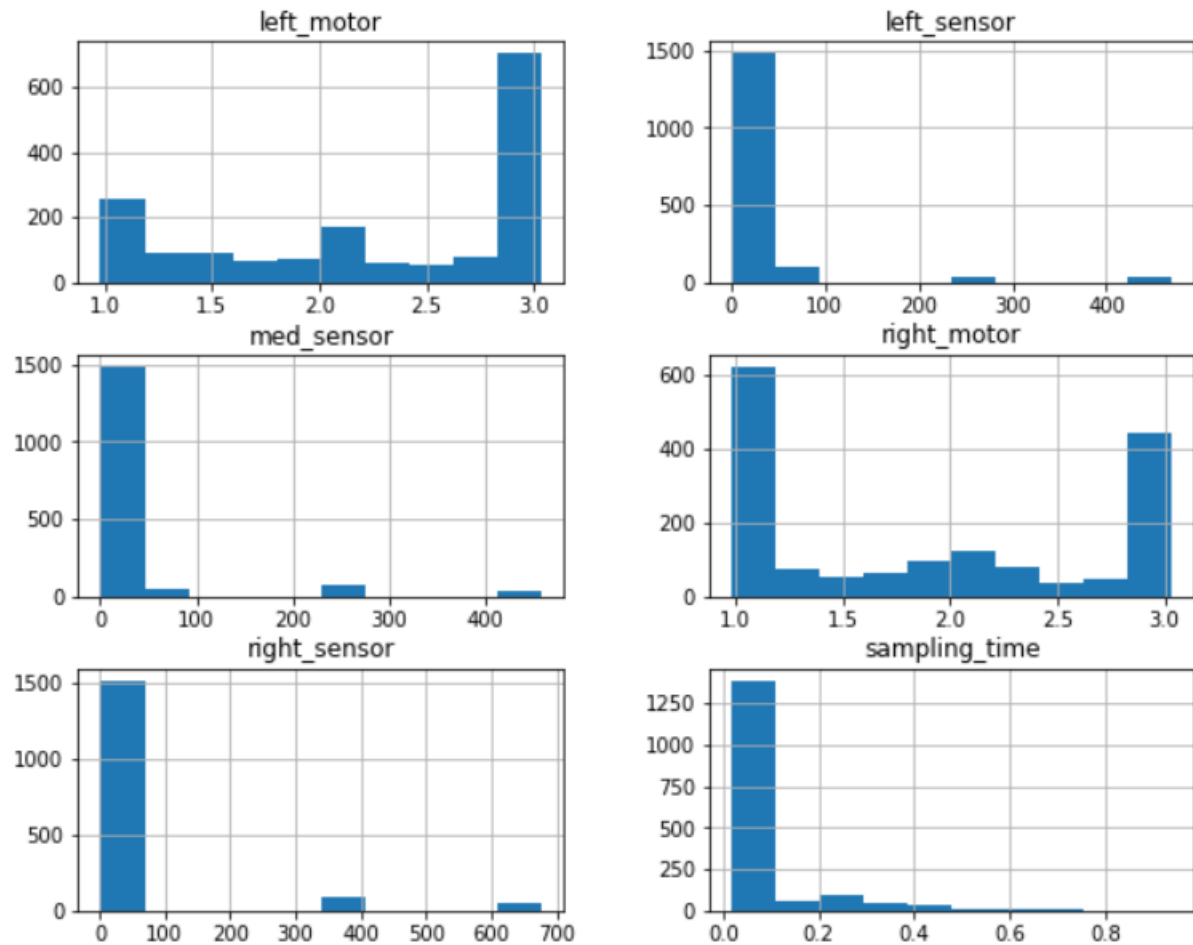


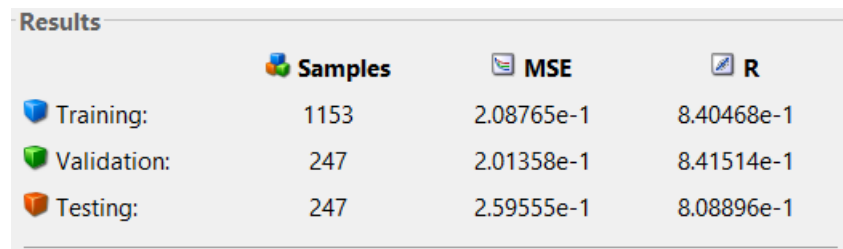
Figure 9: Statistical Distribution of Variables

From above distributions, we can pinpoint the following observations:

- All sensors are giving measurements at some point in time that exceed their maximum detecting range (e.g. 700 cm), this may need recalibration of sensors.
- The sampling time is skewed to the right slightly which is not favorable
- By looking at right motor and left motor distributions, we can generally conclude that the path is circular, and the controller can stabilize the performance which emphasize the rules

NEURAL NETWORK ANALYSIS

In this section of the report we present a Neural Network based analysis to the gathered data for further understanding of the system and its performance. To do this analysis, we used the MATLAB Neural Network Toolbox to create a neural network with one hidden-layer and 7 neurons. Out of the total volume of the data, 70% were used as the training set, and 15% were used for both validation set and testing set. Before confirming the results, we repeated the training process for 10 times. The following figure shows the general results.









	 Samples	 MSE	 R
 Training:	1153	2.08765e-1	8.40468e-1
 Validation:	247	2.01358e-1	8.41514e-1
 Testing:	247	2.59555e-1	8.08896e-1

Figure 10: NN Analysis General Results

We can note the testing set MSE is approximately 0.3 with a probability of 81%. These results are quite good; however, we need to consider the general performance and the number of epochs. The following figure shows the performance.

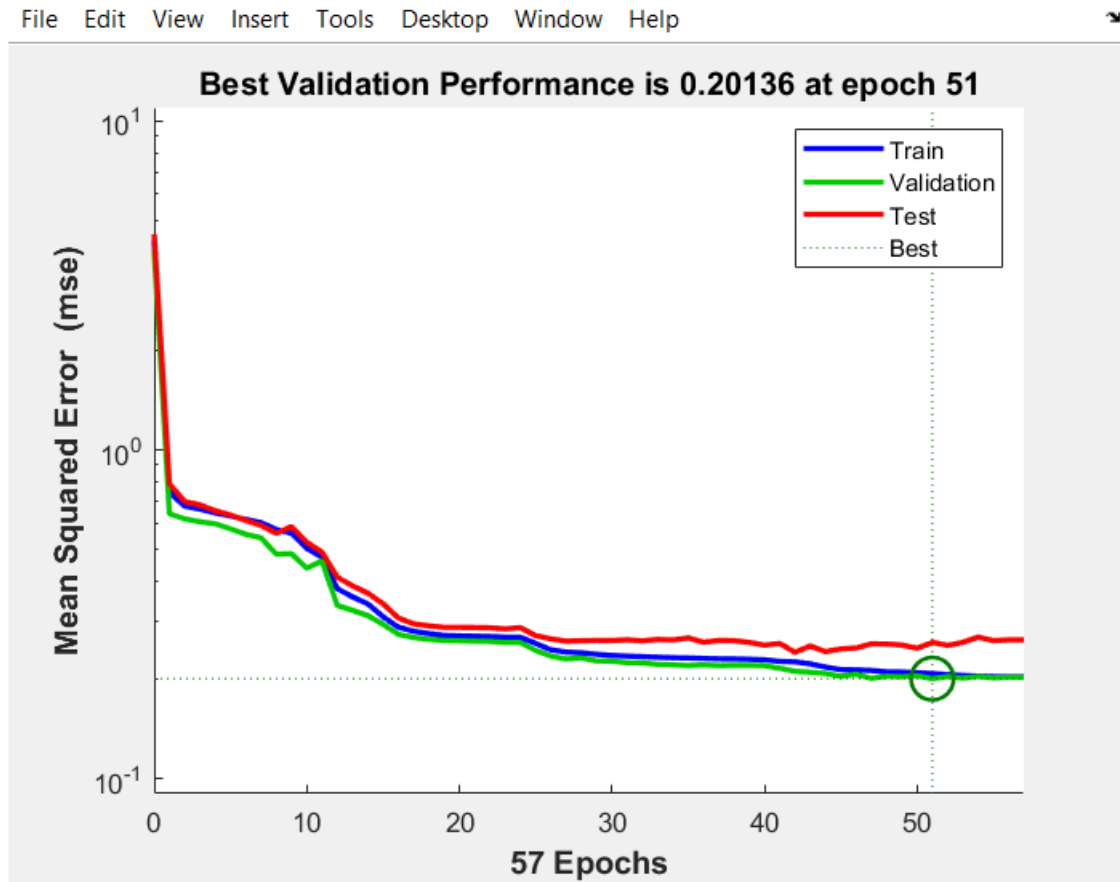


Figure 11: The NN Performance Analysis

It took the model 51 epochs to reach the minimal accuracy level, and the minimum validation error was 0.2014. Thus, this model is reputable and can be implemented further on the hardware directly.

On top of that, the following figure shows an anticipated error distribution appearing to be normal.

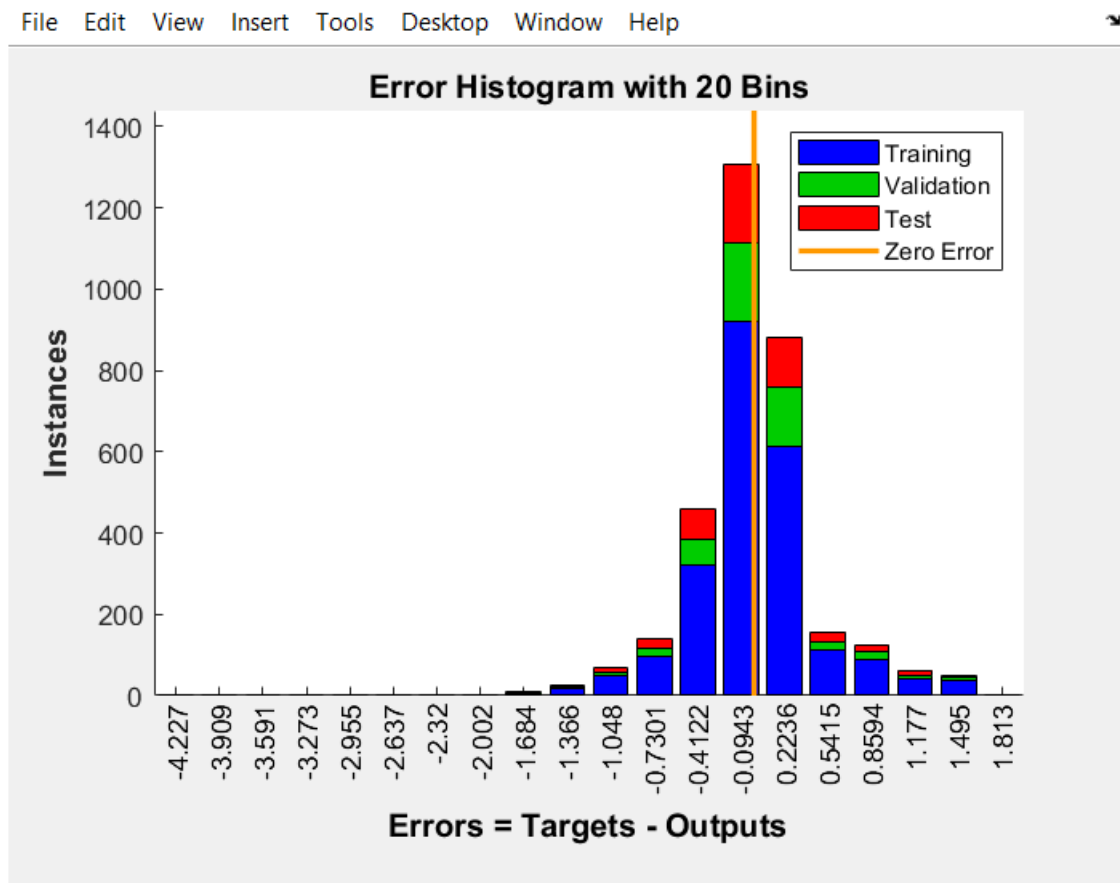


Figure 12: NN Error Distribution

DEMO

We have uploaded a demo video on YouTube to physically illustrates the function of the robot. The following link opens the video.

<https://youtu.be/cokAbExOp3k>

CONCLUSION

The applications of Artificial Intelligence on control can provide rigid designs of robotics, as this report illustrates, which can increase human life quality. In future steps, we aim to implement the NN model of the hardware and compare the result to that of the fuzzy. Adding more functions to the robot is, also, another intention to do shortly. Several functions can be added, for instance, using Cameras to identify paths and direction, carrying weight to other places, etc.

REFERENCES

- Adafruit Website <www.adafruit.com>
- MATLAB Neural Network Toolbox
<<https://www.mathworks.com/help/deeplearning/ref/nnstart.html>>
- Pandas Documentation <www.pandas.pydata.org/pandas-docs>
- Raspberry Pi Website <[www. raspberrypi.org/](http://www.raspberrypi.org/)>